****

**Path Finding Maze Solver**

**MINI PROJECT USING DATA STRUCTURES AND OPERATING SYSTEM**

**(18MCA26)**

**REPORT**

***Submitted by***

**Satyendu Panda**

**1NZ19MCA23**

***In partial fulfillment for the award of the degree of***

**MASTER OF COMPUTER APPLICATIONS**

**2019-2020**



Ring Road, Near Marathahalli,

Bangalore – 560 103

**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS**

**CERTIFICATE**

This is to certify that **Satyendu Panda,** bearing USN **1NZ19MCA23**has successfully completed his second semester mini project work entitled **Path Finding Maze Solver** as a partial fulfillment of the requirements for the award of **MASTER OF COMPUTER APPLICATIONS** degree, during the Academic Year **2019-20** under my supervision. This report has not been submitted to any other Organization/University for any award of degree.

**Signature of the Guide Head of the Department**

**External Viva**

Internal Examiner External Examiner

Date:

**DECLARATION**

I Satyendu Panda student of II Semester MCA, bearing USN **1NZ19MCA23** hereby declare that the project work entitled **“Path Finding Maze Solver”** has been carried out by me under the supervision of Internal Guide **Prof. Jincy C Mathew** **, Assistant Professor** and submitted in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications by Department of Master of Computer Applications, New Horizon College of Engineering, an Autonomous Institution, Affiliated to Visvesvaraya Technological University during the academic year **2019-20**. This report has not been submitted to any other Organization/University for any award of degree.

Name:

Signature: Date:

PLAGIARISM CERTIFICATE

**ACKNOWLEDGEMENT**

I would like to thank Dr. Mohan Manghnani, Chairman of New Horizon College of Engineering for providing good infrastructure and Hi-Tech lab facilities to develop and improve student’s skills.

I sincerely express my gratitude to the college Principal Dr. Manjunatha for supporting the students in all their technical activities and giving guidance to them. I would like to thank Dr. V. Asha, HoD, Department of MCA, New Horizon College of Engineering for granting permission to undertake this project. I would like to express my gratitude to the project guide **Prof. Jincy C Mathew** for giving all the instructions and guidelines at every stage of the Project work.

I thank all the staff members of the Department of Master of Computer Applications, for extending their constant support to complete the project. I express my heartfelt thanks to my parents and friends who were a constant source of support and inspiration throughout the project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter**  **No.** | **Title** | **Page No** |
|  | **LIST OF TABLES**  **LIST OF FIGURES** | **(i)**  **(ii)** |
| **1**  1.1  1.2  1.3  1.4 | **INTRODUCTION**  General Introduction  Project Description  Existing System  Proposed System with Methodology | **1** |
| **2**  2.1 | **REVIEW OF LITERATURE**  Review Summary | **3** |
| **3**  3.1  3.2 | **SYSTEM CONFIGURATION**  Hardware requirements  Software requirements | **15** |
| **4**  4.1  4.2 | **MODULE DESCRIPTION**  Module 1  Module 2 | **16** |
| **5**  5.1 | **SYSTEM DESIGN**  DFD Diagrams | **17** |
| **6**  6.1  6.2 | **SYSTEM IMPLEMENTATION**  Implementation  Screen Shots | **18** |
| **7**  7.1 | **SYSTEM TESTING**  Test Cases | **30** |
| **8**  8.1  8.2 | **RESULTS AND DISCUSSIONS**  Conclusion  Future Enhancements | **32** |
| **9**  9.1  9.2 | **REFERENCES**  Text Reference  Web Reference | **33** |

**LIST OF TABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Table No** | **Title** | **Page No.** |
| 1 | 1 | Hardware Requirement |  |
| 2 | 2 | Software Requirement |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Figure No.** | **Title** | **Page No.** |
| 1 | 1 | Maze template used in training and this Experiment | 5 |
| 2 | 2 | Maze trail | 7 |
| 3 | 3 | Calculation of initial target and optimal path | 9 |
| 4 | 4 | Analyzing the target path | 10 |
| 5 | 5 | Sample Maze | 14 |
| 6 | 6 | Data Flow Diagram no.1 | 17 |
| 7 | 7 | Data Flow diagram no. 2 | 17 |
| 8 | 8 | Screen Shot 1 | 21 |
| 9 | 9 | Screen Shot 2 | 22 |

**ABSTRACT**

Maze solving problem is a very old problem. This field is based on decision making algorithms. Two simple mazes solving algorithms “Wall following algorithm” and “Flood fill algorithm”. In this project, software development and maze construction had been done. Capability of finding the shortest path is also verified. But in this project only “Wall Following algorithm is used to solve the maze in shortest possible way.

**Chapter 1**

**Introduction**

This chapter is going to provide an introduction about this project. These following topics have been described in this chapter, General information about Path Finding Maze Solver, Project Description, Existing System, Proposed System with Methodology and Review of Literature.

**1.1 General Introduction**

A maze may be a network of paths, typically from an entrance to exit. The concept of Maze approximately thousand years old. Which was invented in Egypt. From then, many mathematicians made various algorithm to unravel the maze.

Now a days, maze solving problem is a crucial field. It is based on one of the most important Algorithm, which is “Decision Making Algorithm”. Cause, the starting path is going to be placed in unknown place, and it requires to possess an honest decision-making capability. There are many sorts of maze solving using various sort of algorithms.

In this project, the system design of Maze solving consist obstacle avoidance ultrasonic sensors then sensors will detect the wall. To solve the maze, this will apply wall following algorithms like left- or right-hand rule. It will also follow the Flood fill algorithm for locating the shortest path.

**1.2 Path Finding Maze Solver**

A path finding Maze solver that have a stating path and a goal/exit. The starting path placed in an unknown place on the maze and it trying to find the goal or exit in a shortest possible path. It is a small self-reliant that can solve a maze from a known starting position to the center area of the maze in the shortest possible time. It also has wall and number of rows and columns to present the maze.

**1.4 Proposed System with Methodology**

For solving a maze, it required some searching technique. Here to solve the maze in a shortest possible way I used searching technique DFS (Depth First Search). Maze also surrounded by walls and it have a starting node and goal node i.e. to get out of the node. It used algorithm called “Wall Following algorithm”. To construct the maze, I used some rows and columns that represent the wall. Basically, it used following technique to solve the maze.

* Wall Following Algorithm
* Depth First Search
* For constructing the maze, it used rows and columns
* It solved the maze in shortest possible way.
* It has two nodes i.e. starting and goal node.
* It used some technique called “Decision Making Technique”.

**Chapter 2**

**Literature Review**

This chapter is going to provide important background information and history about the Maze solving robot. Many important theories, method and algorithms on maze solving problem is also given there. This section will start with providing some research about Wall Following Algorithm and Fill Flood Algorithm.

**2.1 Background History-**

In the middle of the 20th century, Maze solving problems become an important field of robotics. In the year of 1972, editors of IEEE Spectrum magazine came up with the concept of micro mouse which is a small microprocessor-controlled vehicle with self-intelligence and capability to navigate a critical maze. Then in May 1977, the fast US Micro mouse contest, called “Amazing Micro mouse Maze Contest” was announced by IEEE Spectrum. From then, this type of contest became more popular, and many types of maze solving robots are developed every year.

Late 1970s the designs of the maze solving robots’ designs were used to have huge physical shapes that contain many blocks logic gates. Figure 1.1 and 1. 2 show the example of early the maze solving robots (micro mouse). Due to technological development the physical size of the robot becomes smaller and the features of the robot becomes modern.

**Analyzing the path of responding in maze-solving and other tasks –**

**About-**

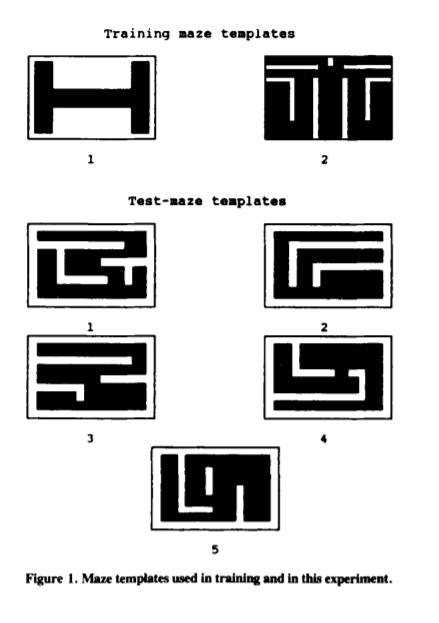
Response time and accuracy are sensitive measures of overall performance but may mask underlying response strategies. For example, analysis of latency and accuracy measures produced in a computerized-maze task does not reveal whether rhesus monkeys really "solve a maze" or simply move as much as is possible toward the target, negotiating barriers through trial and error as they encounter them. Regression procedures are described for analyzing response path against several hypothetical response curves, and analyses of response path for rhesus monkeys' performance on the computerized MAZE task are presented as an illustration. The data suggest that rhesus monkeys do invoke a response strategy of solving the maze, because the observed response topography is significantly associated with the optimal path of responding. Many experimental paradigms should similarly benefit from analysis of the response paths that subjects exhibit.

**Introduction**

Response time (RT) and accuracy are measures that are sensitive to overall performance levels but may mask the response strategies that underlie these levels. For example, the pattern swum by a rat in a water maze, the path of a chimp foraging in a field, and the trail of a cursor as it is moved to computer-generated stimuli on a screen have typically been ignored in favor of more traditional and accessible approximations via RT and accuracy measures. Response paths, however, may contain sensitive indices of the strategies that subjects use for responding.

**ANALYSIS OF RESPONSE PATH**

The general strategy being advanced is one of comparing the observed path of responding-the actual stream of x,y coordinates in time that capture each trial-with one or more hypothetical paths, such as the predictive path or the tracking path in the Washburn and Rumbaugh (in press) study. These hypothetical paths should be derived from theory, and take the form "given a cursor location of Xc, Yc and a target location of Xp,yp, where would the subject move if using strategy SI? Where would the subject move if using strategy Sl? Where in fact did the subject move?"



**Method**

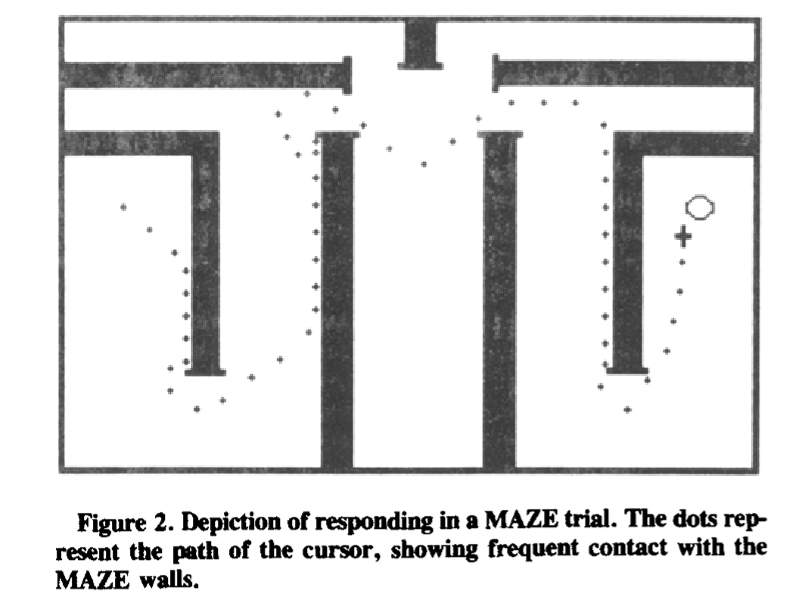
**Subjects**

Ten rhesus monkeys (Macaca mulatta), ranging in age from 2 to 9 years, were tested. Only I of the monkeys was a female. Two of the animals, Abel and Baker, have extensive test histories that have been documented elsewhere (e.g., Rumbaugh, Richardson, Washburn, Savage-Rumbaugh, & Hopkins, 1989; Washburn, Hopkins, & Rumbaugh, 1991). The remaining animals were trained with the procedures and tasks that had been developed for Abel and Baker.

**Apparatus**

**Hardware**. The monkeys were trained and tested with the Language Research Center's Computerized Test System, which consists of a battery of software tasks and the computer hardware required to administer them (see Washburn & Rumbaugh, 1992). An XT-compatible microcomputer recorded all data and presented all stimuli on a color monitor. Each subject responded to these computer-generated stimuli by manipulating a joystick, which in tum controlled the movements of cursor (a l-cm white "+”) on the screen.

**Software**. The task used in the present experiment was the MAZE task, in which subjects had to negotiate computer-graphic barriers on the screen in order to bring the cursor into contact with a small, stationary target (a blue square approximately I cm in size).' The mazes used in this experiment, depicted in Figure I, were drawn with PC-Storyboard and saved as bit-mapped (.PIC) files; they filled the computer screen completely.



**Procedure**

**Training**. Prior to the beginning of this experiment, each subject was trained to respond in the MAZE task with the use of the two training templates depicted in Figure I. In this training, the cursor and target were located randomly within one of the two training mazes. The subjects were trained on each maze until the target was contacted in less than 5 sec, averaged across 200 trials.

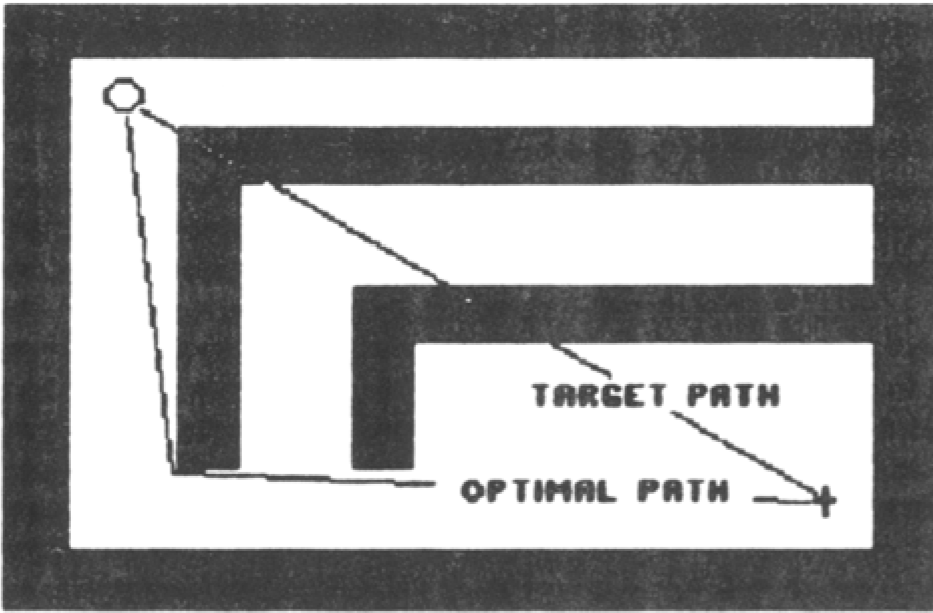
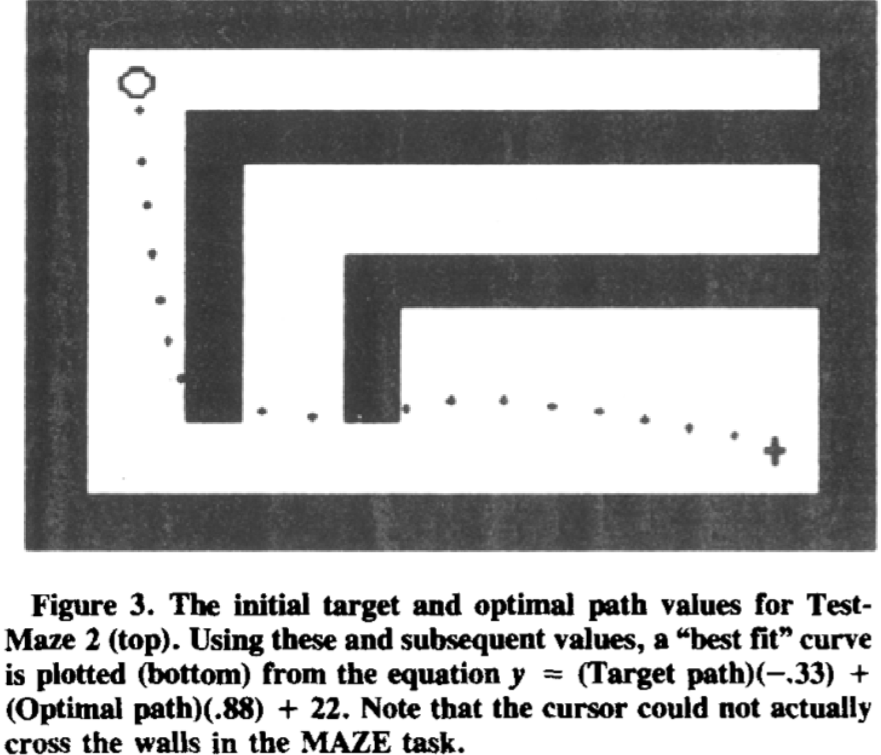
During this training period, each subject learned that maze walls were barriers that could not be traversed, but rather had to be circumvented to reach the target. However, even skilled subjects were observed usually to collide with intervening maze walls en route to the target (see an example in Figure 2). Several potential reasons for this behavior were posited. Subjects might have collided with maze walls in an attempt to minimize travel distance around them. Alternatively, subjects might have exercised the general strategy of moving directly toward the target, and finding a way around the barriers only when absolutely necessary.

**Testing**. The five test-maze templates in Figure I, selected because they cannot be solved by moving directly toward the target, were used in this experiment. Each animal was given a block of 100 trials on each of the mazes, although the order of maze administration was randomized between animals. The cursor always began each trial the lower right corner of the maze, with the target located in the upper left comer, irrespective of which maze template was being tested (i.e., the MAZE task was used in Set-up mode).

In addition to standard RT data, the computer maintained the coordinate-by-coordinate record of the cursor's movements for each trial. Trials that were dropped out were not counted against the required number for completion, although the response path taken on these trials was recorded and used in analysis. Thus, each "path data" file contained all of the information required to reconstruct every response made by each subject for each trial.

**Analysis**. At least two hypothetical response strategies (or algorithms for generating hypothetical paths) could be devised for the MAZE task, against which the observed path of responding was analyzed. First, the subject might attempt to move directly toward the target at each opportunity, moving randomly when absolutely prevented from advancing toward the target (the "target path"). Alternatively, the subject could move along the path of optimal maze solution, without direct regard for the location of the target (the "optimal path"). Of course, other theory-derived paths and combinations of algorithms could be hypothesized, but for the purposes of illustrating the analysis of interest, these two hypothetical paths will suffice.

For analysis, each x, y value, representing each movement of the cursor, was first translated into a single value. Any single unit of measure, such as a distance (using the Pythagorean theorem) or angle, could be used with comparable results. Because cursor movements are a reflection of joystick angle of deflection, the angle of movement was computed for each x, y coordinate in the present analysis [using the formula: angle = (arctangent of the slope). 180111', where the slope = rise/run; see Appendix]. Thus, for each movement during a trial, the angle at which the cursor moved on the screen was computed (relative to the horizon, so that moving directly left was coded as 0°, moving straight up was computed to 90°, and so forth). This angle of movement was compared with (I) the angle that would have been taken if the subject had moved directly toward the target, and (2) the angle that would have been taken if the subject had moved toward the optimal solution of the maze. For each move on the screen, then, three different angles were computed, reflecting a movement-by-movement comparison of potential response strategies.



**Fig 4. Analyzing the Target path with the help of optimal path.**

**RESULTS**

For each of the 500trials per animal in this experiment, an average of109.5 data points (i.e., moves, which were recoded as angles of movement) were generated. Consequently, over one-half million total data points were available for analysis from this sample. Because substantial learning occurred across the 100trials that each animal experienced with a particular maze pattern (revealed by the response times as well as response paths), results of analyzing only the first 10correctly completed trials with each maze will be discussed.

The observed paths of responding were observed to be highly correlated with the two hypothetical paths in this experiment (robserved,target,optimal = .91, P < .01). Thus, over 80%ofthe variance in observed response path was associated with target-directed and optimal-solution strategies. The significant majority of this association, however, was uniquely associated with the hypothetical "optimal" path. The semi partial correlation coefficient for observed and optimal paths was robserved,(optimal.tarlet) = .58(p < .01); the degree to which response topography was uniquely associated with the hypothetical target path was only robserved,(target.optimal) = .05 (p > .05).

**DISCUSSION**

These data suggest that rhesus monkeys tend to respond along the optimal path of responding in simple mazes rather than utilize a basic strategy of always moving toward the target. Collisions of the cursor and the maze wall appear to result from imprecision in joystick manipulation along this optimal path rather than from attempts to exercise a direct route to the target-the present analyses do not reveal a significant unique association between cursor movements and target location. Of course, each of the present mazes afforded a fairly obvious solution, requiring little planning. Further manipulation of the complexity of maze problems (e.g., increasing the number and location of walls) and the complexity of the potential hypothetical paths (e.g., combining strategies and including distance from the walls as a factor) should permit identification of how far in advance rhesus monkeys and other species can "plan ahead" to solve maze problems.

**Chapter 3**

**SYSTEM CONFIGURATION**

In this chapter tells that, what type of hardware and software system required to do the project.

**3.1 Hardware requirements**

|  |  |
| --- | --- |
| Processor | Pentium IV |
| Memory | 512 MB RAM |
| Keyboard | 105 Keys |
| Monitor | CRT or LCD |
| Mouse | USB Mouse or PS/2 Mouse |

**Table 1.0 (Hardware Requirements)**

**3.2 Software requirements**

|  |  |
| --- | --- |
| Operating System | Windows XP/NT |
| Backend End | None |
| Front End | C++ Software |
| Language Used | C Programming |

**Table 1.1 (Software Requirements)**

**Chapter 4**

**MODULE DESCRIPTION**

**4.1 Get Maze Module**

Inget maze module, it first triesto find the maze file that stored in the file named maze.txt. Like this

12,20

+++++++++++++++++++++++++

+ +

+ + + +++++++ + +++++++ +

+ + + +++ ++++ + + ++ + +

+ + + + ++ + +

+ ++++ ++++ + +++ +++++ +

+ s+ ++ +g+ +

+ ++++++++++++ ++ ++++ +

+ + ++ + + + +

+++++++++++++++ +++++++ +

+ +

++++++++++++++++++++++++++

**Fig 5. Sample Maze**

If it finds the maze file then it opened the file in read mode and started reading the first rows first until it finds the End of File (EOF) (In above Fig. 12 is the rows and 20 is the columns).

**4.2 Print Maze Module**

A Maze is given as N\*N binary matrix of blocks where source block is the upper left most block i.e., maze [0][0] and destination block is lower rightmost block i.e., maze[N-1] [N-1]. A rat starts from source and has to reach the destination. The rat can move only in two directions: forward and down.  
In the maze matrix, 0 means the block is a dead end and 1 means the block can be used in the path from source to destination. Note that this is a simple version of the typical Maze problem. For example, a more complex version can be that the rat can move in 4 directions and a more complex version can be with a limited number of moves.

**Chapter 5**

**SYSTEM DESIGN**

The System Design Document is a required document for every project. It should include a high-level description of why the System Design Document has been created, provide what the new system is intended for or is intended to replace and contain detailed descriptions of the architecture and system components.

**5.1 DATA FLOW DIAGRAM**

A data Flow Diagram or a bubble chart is a graphical tool for future analysis. DFD modules a system by using external entities from which data flows to a process, which transforms the data and creates output data flows which go to other process or external entities or files. Data in files may also flow to processes as inputs.

**FLOW DIAGRAM**

Start

Start

Set Origin

Detect

Function

Detect

Function

Move Forward

NO NO

Move Forward

Yes

Calculate Shortest Path

While Object detect

Detect Next

Node

Node

Set Origin

Detect Next

Node

Empty Node

Check

Node

No Obstacle

Backtrack

END

Calculate

Shortest Path

END

**Fig 6. DFD 1 Fig 7. DFD 2**

**Chapter 6**

**SYSTEM IMPLEMENTATION**

Implementation is the stage where the theoretical design is turned into a working system. The system can be implemented only after through testing is done and if it is found to work according to the specification. The implementation phase comprises of several activities. The hardware and software requisition are carried out. Implementation of a computer system is by replacing a manual system. The problem encountered are converting files, training users, creating accurate files, and verifying printouts for integrity.

**6.1 Implementation**

void get\_maze(char\* file\_name)

{

char c;

char rows\_s[3] = { '\0' };

char cols\_s[3] = { '\0' };

int rows\_i = 0;

int cols\_i = 0;

int swap = 0;

FILE\* maze\_file = fopen(file\_name, "r");

if (maze\_file) {

while ((c = getc(maze\_file)) != EOF) {

if (c == '\n') {

break;

} else if (c == ','){

swap = 1;

} else if (!swap) {

rows\_s[rows\_i] = c;

rows\_i++;

} else {

cols\_s[cols\_i] = c;

cols\_i++;

}

**Maze.txt**

12,20

+++++++++++++++++++++++++

+ +

+ + + +++++++ + +++++++ +

+ + + +++ ++++ + + ++ + +

+ + + + ++ + +

+ ++++ ++++ + +++ +++++ +

+ s+ ++ +g+ +

+ ++++++++++++ ++ ++++ +

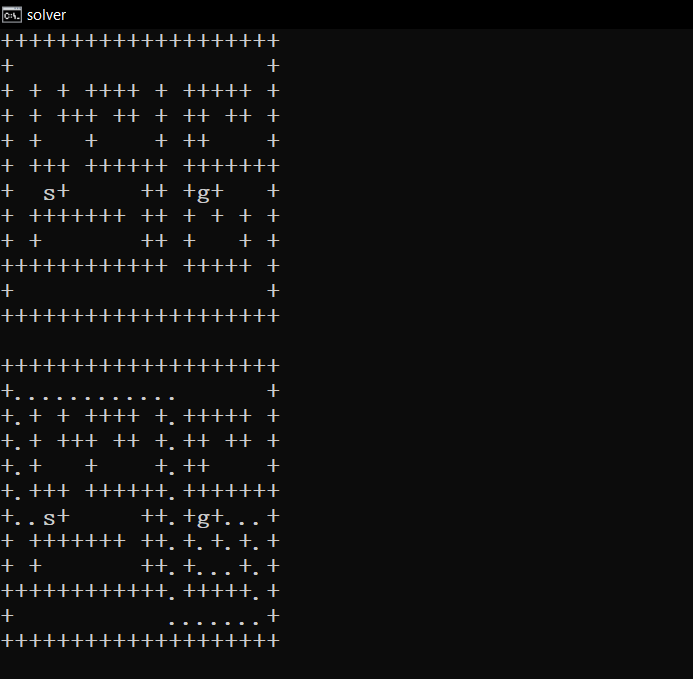
+ + ++ + + + +

+++++++++++++++ +++++++ +

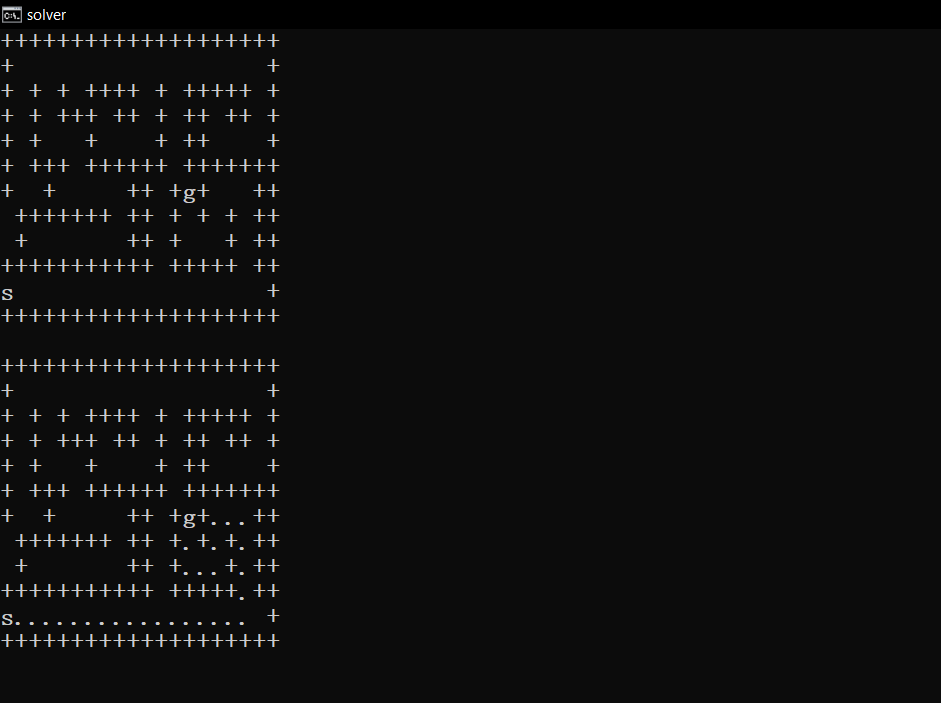
+ +

++++++++++++++++++++++++++

**6.2 SCREEN SHOTS**

****

**Ss.1 It started searching goal node(g) from starting node(s). It must assume all the shortest path before heading further. If it finds the shortest path to goal node then it executed.**

****

**Ss. 2 In this output the starting node(s) changed but the goal node(g) same as previous output. Here it also started searching the shortest possible way to reach the goal node(g). As it found the goal node it executed. We can give different starting node to reach the goal and also, we can give different goal node.**

**Chapter 7**

**SYSTEM TESTING**

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

**Types of testing: -**

**1. Unit Testing**

It focuses on smallest unit of software design. In this we test an individual unit or group of interrelated units. It is often done by programmer by using sample input and observing its corresponding outputs.

**2.** **Integration Testing**

The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components are combined to produce output.

**(a) Black Box Testing:** It is used for validation. In this we ignore internal working mechanism and focus on what is the output?

**(b) White Box Testing:** It is used for verification. In this we focus om internal mechanism i.e. how the output is achieved?

**3. Regression Testing**

Every time new module is added leads to changes in program. This type of testing makes sure that whole component works properly even after adding components to the complete program.

**4. Alpha Testing**

This is a type of validation testing. It is a type of acceptance testing which is done before the product is released to customers. It is typically done by QA people.

**5. Beta Testing**

The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for the limited number of users for testing in real time environment.

**6. System Testing**

In this software is tested such that it works fine for different operating system. It is covered under the black box testing technique. In this we just focus on required input and output without focusing on internal working.

In this we have security testing, recovery testing, stress testing and performance testing.

**7. Stress Testing**

In this we give unfavorable conditions to the system and check how they perform in those condition.

**8. Performance Testing**

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test speed and effectiveness of program.

**Chapter 8**

**RESULTS AND DISCUSSIONS**

**8.1 Conclusions**

As a conclusion, the two mazes solving algorithm have successfully been implemented and the objectives of the project have been achieved. The first algorithm was wall following algorithm. The basic method shows a good result for solving the maze. But, due to lack of self-intelligence, it failed to solve the maze in the shortest way. And it could not solve to close loop maze. So, an efficient method has been used to find the shortest path that is flood fill algorithm method. After applying all methods, the robot was trained in a real maze. Several tests have been run to ensure the best performance of the robot.

This project helps to improve various important information about robotics, knowledge about many decision-making algorithms. It’s also helped to learn about many electronics components such as motor driver, sensors, etc. This gained knowledge will have a significant impact on future work.

**8.2 Future enhancements**

The application was designed in such a way that future modifications can be done easily. The limitations can be improved by adding more modules. As it gives only the basic Maze solving, it could provide some more information too.

It also helped in design of maze solving robot which has the ability to navigate automatically in an unknown area based on its own decision is presented.

**Chapter 9**

**REFERENCES**

**9.1 Text Reference**

[1] DAVID A. WASHBURN Sonny Carter Life Sciences Laboratory, Georgia State University, Atlanta, Georgia “Analyzing the path of responding in maze-solving and other tasks”.

Behavior Research Methods, Instruments, &: Computers 1992, 24 (2), 248-252

[2] W. Matthews, Mazes and labyrinths, 1st ed. New York: Dover Publications, 2013.

[3] A. Bakarsayutisaman and I. Adriane, “Solving a Reconfigurable Maze using Hybrid Wall Follower Algorithm,” International Journal of Computer Applications, vol. 82, no. 3, pp. 22–26, 2013.

[4] M.M Thu, N.N. Win. “Micromouse Maze Solving”, International Journal of Science, Engineering and Technology Research (IJSETR) Volume 5, Issue 9, September 2016.

[5] D. Southall and D. M, "Modified flood algorithm for a maze solving robot ‘mouse’", 2007.

[6] Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory, Sadik, A.M.J. International Conference on Artificial Intelligence and Computational Intelligence, 10.1109/AICI.2010.18, pp.52-56

[7] B. Clough, "Rooter - A MicroMouse Maze Solving Robot", 2007.

[8] R. Zheng, R. Hill and M. Gardener, Engaging older adults with modern technology, 1st ed. Hershey, PA: Information Science Reference, 2013.

[9] "Maze Solvers Archives - cyberneticzoo.com", cyberneticzoo.com, 2017. [Online]. Available: http://cyberneticzoo.com/category/mazesolvers/. [Accessed: 12- Apr- 2017].

[10] "LM7805 Datasheet, PDF - Alldatasheet", Alldatasheet.com, 2017. [Online]. Available: http://www.alldatasheet.com/view.jsp?Searchword=Lm7805. [Accessed: 14- Apr- 2017].

**9.2 Web** **Reference**

[1] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.4944&rep=rep1&type=pdf

[2] https://www.sciencedirect.com/science/article/pii/S1877050917302107

[3] https://www.geeksforgeeks.org/rat-in-a-maze-backtracking-2/

[4] http://arcbotics.com/lessons/maze-solving-home-lessons/

[5] https://ieeexplore.ieee.org/document/5591159/

[6] https://link.springer.com/article/10.3758/BF03203502

[7] https://www.codeproject.com/Articles/5254765/Lee-Algorithm-Mazesolver

[8] https://www.researchgate.net/publication/311674408\_Using\_Differential\_Wheeled\_Robot\_for\_Complex\_Maze\_Solving

[9] https://slideplayer.com/slide/6871473/

[10] http://web.stanford.edu/class/cs106b/assignments/assign2/maze